Indexing Images in Oracle8i

Melliyal Annamalai Rajiv Chopra Samuel DeFazio
Susan Mavris

Oracle Corporation
New England Development Center
One Oracle Drive, Nashua, NH 03062
{mannamal,rchopra,sdefazio,smavris}@us.oracle.com

Abstract

Content-based retrieval of images is the ability to retrieve images that are similar to a query image. Oracle8i Visual Information Retrieval provides this facility based on technology licensed from Virage, Inc. This product is built on top of Oracle8i interMedia which enables storage, retrieval and management of images, audios and videos. Images are matched using attributes such as color, texture and structure and efficient content-based retrieval is provided using indexes of an image index type. The design of the index type is based on a multi-level filtering algorithm. The filters reduce the search space so that the expensive comparison algorithm operates on a small subset of the data. Bitmap indexes are used to evaluate the first filter resulting in a design which performs well and is scalable. The image index type is built using Oracle8i extensible indexing technology, allowing users to create, use, and drop instances of this index type as they would any other standard index. In this paper we present an overview of the product, the design of the image index type, and some performance results of our product.

1 Introduction

The growth of the internet and multimedia technology has led to an increased demand in the market for content processing and management of image data. Some examples of applications that manage image data are e-commerce, museum archives, art gallery archives, digital libraries, medical imaging, multimedia document archives, multimedia publishing, geographic information systems, etc. A large collection of images requires an efficient mechanism for search and retrieval. Keyword based retrieval, where some keywords are chosen to represent the image, does not satisfy the needs of all applications. *Content-based retrieval*, where a user can search for images that are *visually similar* to a given image, is a flexible and accurate method of searching images. The Oracle8i Visual Information Retrieval product provides this facility based on technology licensed from Virage, Inc. [Bac96, GJ97]. It additionally provides a new index mechanism for fast retrieval [Ovr97, Ora98c]. Matching images involves matching high dimensional vectors, and their indexing has been shown to be a difficult problem [Ang95, Fal94, Boz97, Fli95, Whi96].

The design of the index is based on a multi-level filter, where the filters operate on an approximation of the high dimensional data which represents the image, and reduces the search space so that the final computationally expensive comparison is necessary for only a small subset of the data. The reduction of the search space depends on selectivity of the filters. One way to increase the selectivity is to increase the number of dimensions used in the filters. Higher dimensions result in more accuracy of comparisons and hence better selectivity. Systems using the multi-level filter approach typically reduce the number of dimensions enough so that the filter can be based on an index structure such as an R-tree, KD-tree, SS-tree, X-tree etc [Fal94, Fli95]. In our approach we implement the first filter as a range

MOD 2000, Dallas, TX USA

query, one range per dimension, which enables us to use bitmap indexes to efficiently evaluate the query. Bitmap indexes are constructed on each dimension. The efficiency of indexing individual dimensions and combining results when using Oracle's bitmap indexes makes it possible to have a high dimensional (83 dimensions in this implementation) first filter which results in high selectivity and scalable performance. We present results for indexing and searching up to 100,000 images.

The extensible architecture of the Oracle server has enabled the addition of efficient and effective technology for the multimedia domain. A special image index type for content-based retrieval has been built using the Oracle8i extensible indexing framework [Ora98a]. This allows encapsulation of the index management routines in an index type schema object. It enables an index which is an instance of this index type to be created and used the same way as any other Oracle Server index. The index-based implementation of the SQL operators VIRSimilar and VIRScore can be used to search for images similar to a given query image. The operators can be used in SQL statements like any other operator.

The Oracle8i Visual Information Retrieval product is built on Oracle8i interMedia, which enables storage, retrieval and management of images, audios, and videos, and manipulation and format conversion of images. All the image functionality of interMedia is supported in this product. In this paper we first present a brief background and our terminology, followed by a description of the SQL interface of our product. We then describe the design of the index type and conclude with a section on issues related to performance.

2 Our Content-based Retrieval Model

The specifics of the content-based retrieval model chosen by Oracle is based on cooperation with Virage. Other technology can be incorporated into Oracle using similar concepts.

Each image in the Visual Information Retrieval product is represented by a high dimensional *signature*. The signature is an abstraction of the contents of the image in terms of its visual attributes. It is represented using 2000 bytes. Two images are compared by comparing their signatures. Comparison of signatures yields a *score* value between 0 and 100 which represents the degree of similarity between the two images. A value of 0 means a perfect match and a value of 100 means the two images are completely different from each other.
The signature contains information about the following visual attributes:

- *Global Color:* This attribute represents the colors in the entire image.
- *Local Color:* This attribute represents the spatial distribution of the colors in the image.
- *Texture:* This attribute represents the patterns in the image such as graininess and smoothness.
- *Structure:* This attribute represents the shapes that appear in the image, as determined by shape characterization techniques.

The user can assign a *weight* or importance measure to each of these visual attributes in a query. Each weight value indicates how sensitive the similarity computation should be to that visual attribute. The concept of weights gives the user flexibility in defining her search.

3 The SQL Interface

The Oracle8i Visual Information Retrieval product adds new operators and datatypes to Oracle's SQL interface.

3.1 Visual Information Retrieval Datatype

Object-relational extensions to Oracle8i can provide support for domain-specific applications. These extensions enable the provision of solution-oriented capabilities unavailable in the server. New datatypes can be defined which are integrated with the server engine so that the optimizer, query parser, indexer and other server components are aware of them. Data stored using these datatypes have access to all database services such as backup, recovery, transaction control, multi-user access etc. The Oracle interMedia product is such an extension that provides image, audio and video datatypes whose

behavior is defined by the attributes and methods included in the product [Ora98b]. The object provided for storing images in interMedia is ORDImage. The Visual Information Retrieval product is layered on top of the interMedia product (Figure 1). ORDImage is extended to form the ORDVIR object by adding an attribute *signature* to the object. ORDImage contains the ORDSource object and content metadata such as height, width, content length, compression format, mime type of the image. ORDSource contains information about the location of the object - in the database, as an external file such as on a CD (but with the content metadata and the signature in the database), or as a URL.

The attributes of the ORDVIR object are:

ORDImage
signature

The attributes of the ORDImage object are:

ORDSource	height
contentLength	width
fileFormat	contentFormat
compressionFormat	mimeType

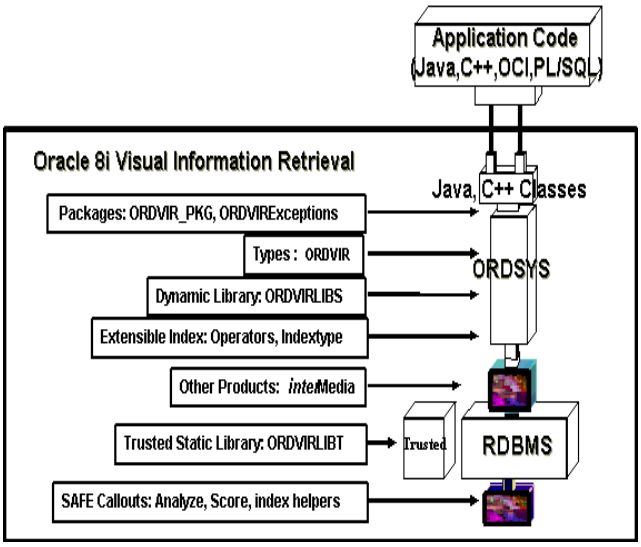

Figure 1: Architecture of the Visual Information Retrieval Product

3.2 Visual Information Retrieval Operators

Two new SQL level operators have been provided in Oracle8i:

1. VIRSimilar(signature_column, query_signature, weightstring, threshold, number).

   - signature_column: is the column of signatures in the table of images
   - query_signature: is the signature of the query image
   - weightstring: is the list of weights used to specify the importance of attributes used in the comparison
   - threshold: is a value which can be used by the user to specify the degree of similarity
   - number: is the parameter which matches the invocation of this operator with that of VIRScore.

   VIRSimilar returns 1 if two images match and 0 otherwise.

2. VIRScore(number)
   VIRScore is an ancillary operator that returns the score of similarity computed by VIRSimilar. It is used in conjunction with VIRSimilar.

   - number: is the parameter which matches the invocation of this operator with that of VIRSimilar.

The index type ORDVIRIDX enables the creation of the image index. The indextype encapsulates all the routines necessary for DDL, DML and query operations of an image index.

3.3 Examples of Usage

The following are examples of creating and using a Visual Information Retrieval index:

Create and populate a table:

```
CREATE TABLE images (id NUMBER, img ORDVIR);
INSERT INTO images VALUES (...);
```

Create index:

```
CREATE INDEX imgidx ON images(img.SIGNATU
INDEXTYPE IS ORDVIRIDX PARAMETERS(.....)
```

Query:

```
SELECT T.id, VIRSCORE(1) FROM images T
WHERE VIRSIMILAR(T.img.SIGNATURE,
querySignature, 'globalcolor=0.5, localcolor=0.0,
texture=0.5, structure=0.0',10,1) = 1;
```

Thus the user can avail herself of the image indexing technique through a straight forward SQL interface and can create and use the index in a manner similar to that of a standard index. Insert, delete and update operations on the underlying table are also performed in exactly the same way as with a standard index. From the user's point of view the usage and behavior of the image index which is an extensible index is the same as that of a standard index.

4 Design of Visual Information Retrieval Index

The Visual Information Retrieval index, unlike other database indexes, is not one single data structure but a collection of database schema objects. It consists of:

- *Feature Table:* Each image has a row in this internal table. Each row contains a set of numbers which are an approximate representation of the signature. (in this implementation there are 83 numbers).
- *Bitmap Indexes:* A set of bitmap indexes are created on columns in the above feature table.

The index lookup consists of three phases (Figure 2). The first two phases are filters, which reduce the search space, and the last phase is where the final matches are performed.

4.1 Phase 1: Evaluation of Filter 1

As we mentioned earlier, several systems use a multi-dimensional indexing structure such as an R-tree, KD-tree, SS-tree or X-tree to index the first filter [Fal94, Fli95]. For these index structures to be effective the number of

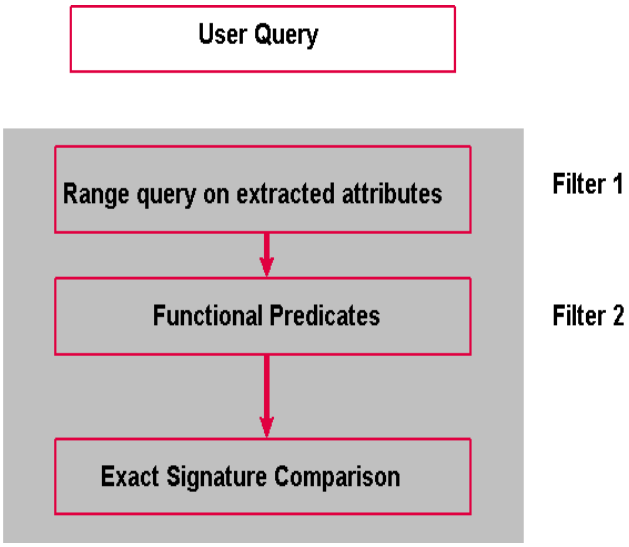

Figure 2: Query Execution Path

dimensions has to be low, in the range of 15-20. They do not scale well beyond these dimensions. QBIC [Fli95] is an example of an approach using the filtering technique. The dimensionality is reduced using K-L transform to get an approximate signature with dimensions as low as 2-3 which can be indexed using R^*-trees. Approximating the high dimensional signatures using low dimensional approximations of the signature typically results in a low selectivity (higher number of images are selected) in the first filter. Thus a higher number of images have to be processed in subsequent filters and in the final signature comparison stage which downgrades performance.

In our approach, we use individual bitmaps to search the approximate representation of the signature. In our current implementation the approximate signature has a high dimension (83 dimensions), and we create 83 bitmap indexes to search these approximate signatures. We identified that the cardinality of most of the dimensions was low, and the advantages of bitmaps are greatest for low cardinality data. Low cardinality means that the number of distinct values is low when compared to the number of rows in the table. This is true for many image attributes, where millions of images are represented by a fixed number of image attributes. Usage of multiple bitmap indexes is an efficient and scalable solution,

because of the efficiency and scalability of Oracle's bitmap index implementation.

4.1.1 Bitmap Indexes

In a regular (B-tree) index a list of rowids for each key is stored. In a bitmap index a bitmap for each key value is used instead of a list of rowids. Each bit corresponds to a possible rowid, and if the bit is set, it means that the row with the corresponding rowid contains the key value [Ora98a].

Bitmap indexes are suitable when: (1) users primarily query the data rather than update it (2) the cardinality of the data is low (3) the data is involved in complex conditions in the WHERE clause in queries. Our design of the first filter satisfies all three conditions.

Queries with AND or OR conditions in the WHERE clause can be quickly resolved by performing the corresponding boolean operations directly on the bitmaps before converting the bitmaps into rowids [Ora98a], contributing to the scalability of the filter. The approximate signatures are stored in a feature table with 83 columns, with each column indexed by a bitmap. The first filter is executed by a query on the feature table. The query is a conjunction of range query terms, one range per column. The results of the individual bitmap index lookups are efficiently merged.

Another advantage of using bitmap indexes is that indexes on dimensions not used for execution of a particular query need not be accessed and used. The 83 columns are grouped according to the four attributes global color, local color, texture, and structure. If only global color and texture weights are specified in the query, only those bitmaps need to be used. This results in less data being read and fewer bitmaps being merged and thus improved performance for such queries. Such a selective dropping of dimensions is not easily possible with other index structures.

A detailed discussion on selectivity of the first filter is presented in the performance section.

4.1.2 Index Size

A typical application scenario would store images in an image table, one image in each row. Each row in the image table has an entry in the feature table. The feature table along with the bitmap indexes created on it has a total size that is not more than 30% of the original table size. The signature of each image in the image table is 2000 bytes, and each row in the feature table is not more than 300 bytes and the size of the bitmap indexes for each row is not more than 300 bytes.

4.2 Phase 2: Evaluation of Filter 2

The rows that pass the first filter in the previous phase are now passed through a second filter. This second filter is a distance measure computation that cannot be indexed or represented easily using SQL and is executed in C. The details are proprietary information to Virage, Inc.

4.3 Phase 3: Signature Comparison

The query signature is now matched with the signatures of the images that passed the second filter. A score is returned for each image and if the score is less than the user-specified threshold the image is returned as a match.

We can thus see that the index is in reality an assembly of database objects (Figure 3). Execution of a query involves execution of several steps accessing several database objects. First a range query which is a conjunction of several clauses is created from the query signature. This query is executed using the bitmap indexes on the feature table. The results of this query, which are the results of the first filter, are passed to the second filter. The final signature comparison using Virage's code is executed for the results of this filter.

4.4 DML Operations

Insert of a new signature is executed by extracting the approximate representation of the signature and adding that row to the feature table, which causes the bitmap indexes to be updated. Deletion of a signature is executed by deleting the row in the feature table which causes the bitmap indexes to be updated. Update of a signature triggers the regeneration of the approximate signature and update of the bitmap indexes.

4.5 Oracle and Virage

Oracle8i server and Virage software modules cooperate to execute the three phases. Virage routines execute in a process space separate

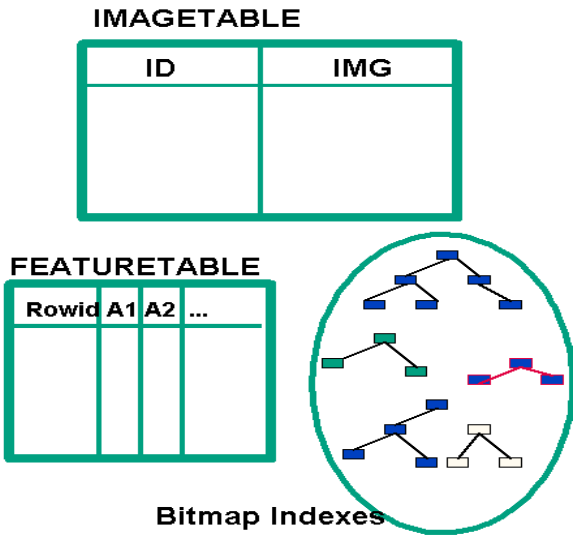

Figure 3: Components of the Image Index

from that of the Oracle server. In phase 1, the range query terms are generated by a Virage function. In phase 3, the final signature comparison is executed by a callout to an external Virage function.

It has to be noted that while the implementation details of this product have been designed for successful interaction between Oracle and Virage, the same concept can be extended to other domain specific products such as face recognition, fingerprint matching, and region-based querying. Oracle's extensible architecture allows any specialized algorithm or product to be incorporated seamlessly in the server.

5 Performance Issues

It is interesting to investigate the various factors that affect the performance of the index. The performance is the combination of the performance of the three phases seen above. Obviously, in a multi-level filtering scheme, only the first filter can be indexed, and the performance of the following phases will be a factor of the number of images they process. The first filter design determines the scalability of the index and the bitmap index approach provides this scalability. The fact that we can drop dimensions which are not used in the match also influences performance.

The dominant factor influencing the performance of the index is the selectivity of the first filter. If the selectivity is high (number of rows returned is low), then the second and third phases operate on only a small percentage of the data, resulting in good performance. If the selectivity is low, performance deteriorates, though it is always linear with a slope less than 1. Some situations where the first filter selectivity is high are - low threshold indicating that the user is interested in only exact or near-exact matches, the query image being dissimilar to most of the images in the database, a selective attribute such as the texture attribute being the only one specified. Our index performs best in these situations, and in the worst case where selectivity crosses 50% the performance scalability is still linear with a slope less than 1.

In this section we present results on the variation of performance with threshold and specified attributes. To nullify the factors of distribution of data in the database and similarity of the image query with images in the database, we create a database populated with generic photographic images with a uniform distribution with respect to the four attributes.

5.1 Performance Setup

The experiments were conducted on a two CPU Ultra Enterprise Solaris machine running SunOS 5.6 and with 1 GB RAM. The image table size ranged from 10,000 rows to 100,000 rows.

Our experiments involved the following queries:

- Varying number of attributes specified: (threshold is 10)
  - ColorTex.sql: Nonzero weights for global color and texture only (the user wants to use only those attributes for the match).
  - Tex.sql: Nonzero weights for texture only.
  - TexStruc.sql: Nonzero weights for texture and structure only.
- Varying threshold: (attributes are global color and texture)
  - Thresh3.sql: The user is interested in only exact or near-exact matches.
  - Thresh10.sql: Threshold of 10.

– Thresh15.sql: Threshold of 15.

– Thresh20.sql: Threshold of 20. The selectivity of the first filter usually deteriorates.

In all our results disk I/O represents 10% of the total query time. CPU time is the dominant factor because of the bitmap merge phase and minimized I/O cost while accessing the bitmaps. The times represent measurements in our environment. The goal is to present the relative performance of the different queries rather than absolute numbers.

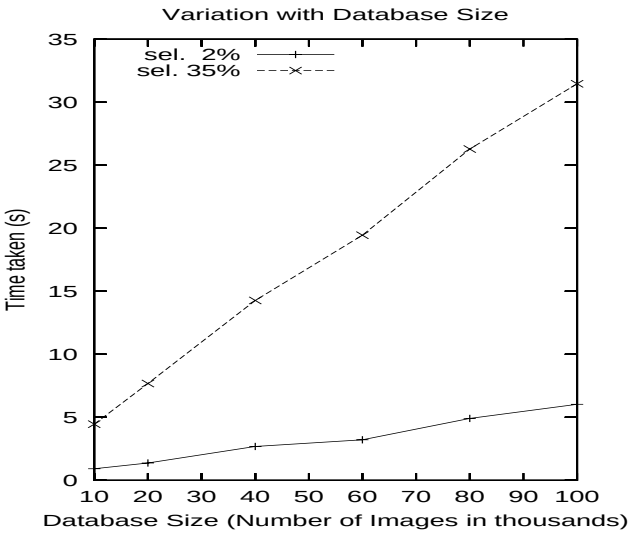

Figure 4: Variation with Database Size

5.2 Discussion

Figure 4 illustrates the performance of the query ColorTex varying with the size of the image table. It can be seen that the performance scales well with size. The figure shows the time taken for two queries each with different query images. The two queries have different first filter selectivities representing both ends of the spectrum - 2% selectivity and 35% selectivity. The time taken by the query with 35% selectivity increases more rapidly with size than the one with 2% selectivity. This is because with decreased selectivity the expensive signature comparison becomes the significant component of the total time, and that time grows more rapidly with time than the first filter time.

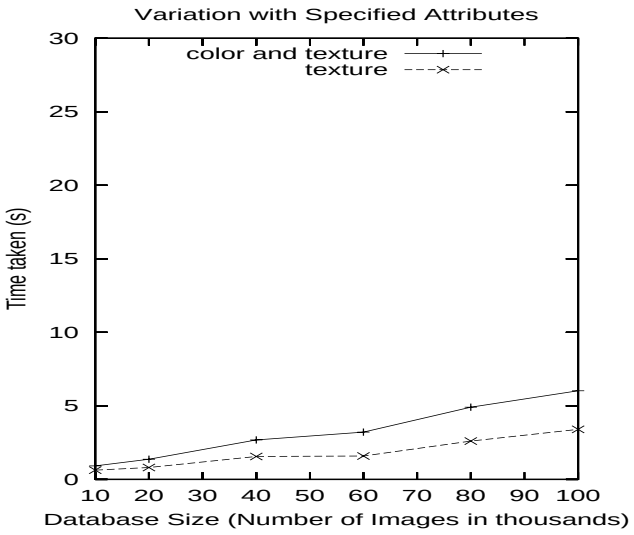

Figure 5: Variation with Number of Attributes

However, as indicated earlier, the slope is less than 1.

Figure 5 illustrates how performance varies with the attributes specified. The texture only query Tex has a higher selectivity than the ColorTex query resulting in better performance for such queries. The better performance is also because only the bitmaps corresponding to the texture attribute are accessed and merged, where as in the ColorTex query bitmaps corresponding to two attributes have to be accessed and merged.

Figure 6 illustrates how threshold affects performance. Queries Thresh3, Thresh10, Thresh15, Thresh20 are executed. When the threshold is 3, only exact or near-exact matches are retrieved. In this situation a large number of images are eliminated in the first filter. As the threshold increases the selectivity decreases and the time taken increases. The selectivity of the first filter when the threshold is 3 is less than 0.5%, when the threshold is 10 its 2%, when the threshold is 15 its 13% and when the threshold is 20 it jumps to 65%.

It has to be emphasized that these measures for time taken and selectivity could vary depending on the image data in a particular database. For instance, if all the images in the database are very similar to each other, these figures will change significantly.

The results clearly illustrate some of the

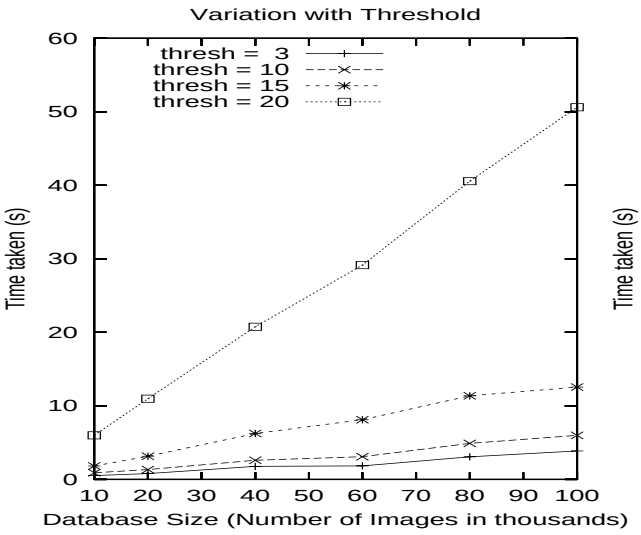

Figure 6: Variation with Threshold

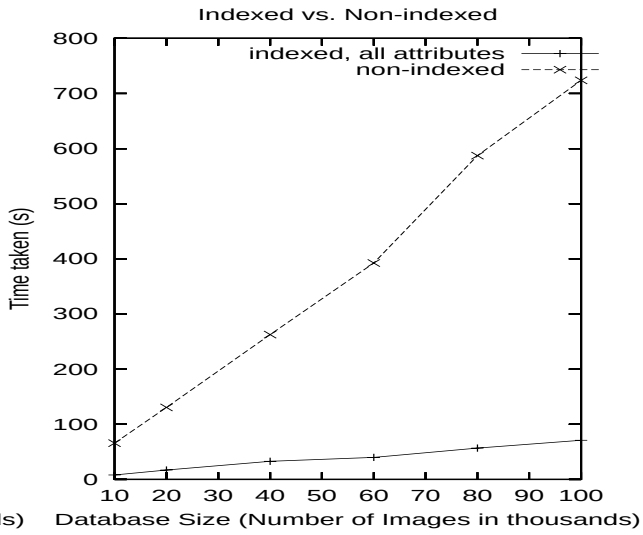

Figure 7: Comparison of Indexed vs. Non-indexed implementation

advantages of our approach. When the user is not interested in certain dimensions, those are not used, leading to lesser bitmap indexes being read and merged, resulting in quicker execution of the query. Similarly when only the exact matches are required, efficient execution is possible. Determining that there are no good matches in the database is also very efficient. Users in some scenarios have a tremendous benefit, users in other scenarios might have a lesser benefit but still experience reasonable performance. These results give us some heuristics for using the index. For instance a large threshold results in a large result set and a linear scan might be better.

5.3 Indexed vs. Non-indexed

Figure 7 illustrates the tremendous improvement in using the index in Oracle. Even when all the attributes are used the indexed implementation outperforms the non-indexed implementation by an order of magnitude. When fewer attributes are used the improvement can be two or three orders of magnitude.

6 Conclusion

The Oracle8i Visual Information Retrieval Product provides for efficient content-based retrieval of images. It provides an image index type using Oracle8i's extensible indexing mechanism which enables users to use an image index just as they would any other Oracle index. The index implementation is based on a multi-level filter, with the first filter based on multiple bitmap indexes. Various factors such as low cardinality of data, primarily querying the data rather than updating it and selection of bitmap indexes that correspond to attributes actually used result in the bitmap indexes providing a scalable solution.

7 Acknowledgements

We thank Jeff Bach and his colleagues at Virage, Inc. for their assistance during the design and implementation of this product.

References

[Ang95] Y. H. Ang, Z. Li, S. H. Ong, Image Retrieval based on Multidimensional Feature Properties, In *SPIE*, Vol. 2420, 47–57, 1995.

[Bac96] J. R. Bach, The Virage Image Search Engine: An Open Framework for Image Management, In *Proceedings of Storage and Retrieval for Still Image and Video Databases IV*, SPIE Vol. 2670, 76–87. 1996.

[Ber98] S. Berchtold, C. Bohm, H-P. Kriegal, The Pyramid-Technique: Towards Breaking the Curse of Dimensionality, In *Proceedings of the 1998 ACM SIGMOD*, 142–153, 1998.

[Boz97] T. Bozkaya, M. Ozsoyoglu, Distance-based Indexing for High-dimensional Metric Spaces, In *Proceedings of the 1997 ACM SIGMOD*, 357–368, 1997.

[Fal94] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic and W. Equitz, Efficient and Effective Querying by Image Content, In *Journal of Intelligent Information Systems*, 3(3), 231–262, 1994.

[Fli95] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele and P. Yanker, Query by Image and Video Content: The QBIC System, *IEEE Computer*, 28(9), 23–31, 1995.

[GJ97] A. Gupta and R. Jain, Visual Information Retrieval, *Communications of ACM*, 40(5):70–79, May 1997.

[Ovr97] Leveraging Digital Image Assets Using the Oracle8 Image Cartridge and the Oracle8 Visual Information Retrieval Cartridge, *Oracle Business White Paper*, June 1997.

[Ora98b] *Oracle8i interMedia Audio, Image, and Video*, Oracle Corporation, February 1999.

[Ora98a] *Oracle8i Server Concepts*, Oracle Corporation, February 1999.

[Ora98c] *Oracle8i Visual Information Retrieval*, Oracle Corporation, February 1999.

[Whi96] D.A. White, R. Jain, Similarity with the SS-tree, In *Proceedings of the 12th International Conference on Data Engineering*, 1996.